Refine Search

Search Results -

Terms	Documents
L22 and L1	4

Database: US Pre-Grant Publication Full-Text Database; US Patents Full-Text Database; US OCR Full-Text Database; EPO Abstracts Database; JPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L23

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Thursday, May 11, 2006 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L23	L22 and l1	4	L23
L22	L12 same position same xo$	54	L22
L21	L20 same l4	3	L21
L20	L19 same ('1' or '0')	336	L20
L19	L18 same register	533	L19
L18	L5 same shift$	883	L18
L17	L15 same l5	0	L17
L16	L15 same l1	2	L16
L15	L14 same l12	164	L15
L14	multi-bit	13391	L14
L13	L12 same l10	10	L13
L12	shifting same bit same register	11301	L12
L11	L10 same l9	5	L11
L10	l1 near3 value	2149	L10

L9	shifting same (significant adj1 bit) same register	1425	L9
L8	shifting	454324	L8
L7	L6 and l3	7	L7
L6	L5 same l4	10	L6
L5	lsb same msb same (significant adj1 bit)	5327	L5
L4	generator adj1 polynomial	1830	L4
L3	L1 same (calculat$ or generat$)	11013	L3
L2	L1 same calculat$	4207	L2
L1	(cyclic adj1 redundancy) or crc	44867	L1

END OF SEARCH HISTORY

L13: Entry 1 of 10 File: USPT Mar 4, 2003

DOCUMENT-IDENTIFIER: US 6530057 B1
**** See image for Certificate of Correction ****
TITLE: High speed generation and checking of cyclic redundancy check values

CLAIMS:

15. A method of calculating a CRC value for a unit of data, comprising: producing, while calculating said CRC value for said unit of data, a first output comprising a first modulo 2 remainder of a division of a subset of said unit of data by a generating polynomial; shifting, while calculating said CRC value for said unit of data, an output of a remainder register by the number of bits in said subset of said unit of data to form a shifted output, and producing a second output comprising a second modulo 2 remainder of a division of said shifted output by said generating polynomial; and performing, while calculating said CRC value for said unit of data, a bit-wise exclusive-OR logic operation on said first output and said second output to produce a third output, said third output coupled to an input of said remainder register.

28. A system for calculating a CRC value for a unit of data, comprising: first means for producing, while calculating said CRC value for said unit of data, a first output comprising a first modulo 2 remainder of a division of a subset of said unit of data by a generating polynomial; second means for shifting, while calculating said CRC value for said unit of data, an output of a remainder register by the number of bits in said subset of said unit of data to form a shifted output, and producing a second output comprising a second modulo 2 remainder of a division of said shifted output by said generating polynomial; and third means, for performing, while calculating said CRC value for said unit of data, a bit-wise exclusive-OR logic operation on said first output and said second output to produce a third output, said third output coupled to an input of said remainder register.

32. A method of checking a CRC value for a unit of data, comprising: producing, while checking said CRC value for said unit of data, a first output comprising a first modulo 2 remainder of a division of a subset of said unit of data by a generating polynomial; shifting, while checking said CRC value for said unit of data, an output of a remainder register by the number of bits in said subset of said unit of data to form a shifted output, and producing a second output comprising a second modulo 2 remainder of a division of said shifted output by said generating polynomial; performing, while checking said CRC value for said unit of data, a bit-wise exclusive-OR logic operation on said first output and said second output to produce a third output, said third output coupled to said remainder register; comparing, while checking said CRC value for said unit of data, said output of said remainder register to a predetermined value; and reporting an error if there is not a match between said output of said remainder register and said predetermined value.

First Hit Previous Doc Next Doc Go to Doc#

End of Result Set

Generate Collection Print

L7: Entry 7 of 7 File: DWPI Oct 7, 2004

DERWENT-ACC-NO: 2004-469882
DERWENT-WEEK: 200466

TITLE: Cyclic redundancy check value calculation method for multi-bit input data in data traffic system, involves shifting repeatedly new XOR-ed bit to most significant bit of register

Basic Abstract Text (1):
NOVELTY - The input word is serially shifted into cyclic redundancy check (CRC) register. The register contents is XOR-ed with generator polynomial, when least significant bit (LSB) of register is 1'. The contents of register is shifted right by one position and new XOR-ed bit is repeatedly shifted into most significant bit (MSB) of register. The number of zeros equal to polynomial length is shifted, to calculate CRC value.

Basic Abstract Text (6):
(4) CRC value calculation apparatus; and

Basic Abstract Text (8):
USE - For calculating CRC value for multi-bit input data word in data traffic system, using defined generator polynomial.

Basic Abstract Text (9):
ADVANTAGE - Performs CRC calculations on data quickly and efficiently.

Previous Doc Next Doc Go to Doc#

L7: Entry 6 of 7 File: USPT May 14, 1996

DOCUMENT-IDENTIFIER: US 5517512 A
TITLE: Cyclic coding and cyclic redundancy code check processor

Abstract Text (1):
A cyclic coding and cyclic redundancy check processor for a generator polynomial G (X) with a degree of m (m is an integer satisfying m.ltoreq.n/2) by units of n bits comprising a residue table storage section to store a residue table containing 2.sup.n pieces of residue data each having n/2 bits obtained by divisions of 2.sup.n pieces of data having n bits by the generator polynomial, a data read section to read out the data having predetermined number of bits for cyclic coding by units of n bits, an address generation section to generate addresses of the residue data to be read out from the residue table storage section, a residue data read section to read the residue data out of the residue table storage section according to the generated address, a processing section for shifting and saving of the data subjected to cyclic coding and a data synthesis section for synthesis of the data shifted or saved and the read out residue data.

Brief Summary Text (6):
On the other hand, CRC is a process carried out at the receiver to detect errors in the cyclic coded data as described above. Conventionally, there have been two types of cyclic coding and CRC processor for computerized cyclic coding and CRC corresponding to the degree of the generator polynomial and the number of data bits subjected to cyclic coding. Since the cyclic coding and CRC basically have the same procedure, the following description mainly addresses the cyclic coding.

Brief Summary Text (8):
A cyclic coding and CRC processor in a first conventional example is effective only when the generator polynomial has a low degree and a small number of data bits are processed. As shown in FIG. 6, it comprises a central processing unit (CPU) 600 for operation, data processing and control with a unit of 8 bits, a data memory 620 which stores 50 bits of the data for cyclic coding in order and a table ROM 610 which stores a syndrome table, all of which are connected with a bus 630.

Brief Summary Text (19):
The above procedure is performed for 8 bits (Step 905). Then, in Step 906, the values of registers A and B are shifted by one bit and the MSB of register B is transmitted to the least significant bit (LSB) of register A and the MSB of register A to the carry register (CY). In Step 907, it is judged whether CY=1 or not. If CY is 1, the values of register A and generator polynomial register are XORed and the result is stored to register A (Step 908). If CY is 0 in Step 907, then the CPU 800 just proceeds to Step 909. Then, the CPU 800 performs such bit judgment processing for the remaining 7 bits (a.sub.48 to a.sub.42) firstly stored to register A (Steps 906 to 910). At this time, the data of register B (a.sub.n to a.sub.n-7 ; n=41) are entirely shifted to register A, and the CPU 800 reads out the following 8 bit data (a.sub.n to a.sub.n-7 ; n=33) from the data memory 820, stores the data to register B and follows Steps 906 to 910. The CPU 800 performs the procedure of Steps 904 to 910 for the number of loops (for 48 bits). Then, in Steps 913 to 919, it performs similar bit judgment for the remaining 2 bits. The higher 3 bits of the value at register A upon completion of processing are the parity bits (Step 920).

Brief Summary Text (31):
According to another preferred embodiment to attain the above objects, a cyclic redundancy check processor for a generator polynomial G(X) with a degree of m (m is an integer satisfying m.ltoreq.n/2) using a computer for data processing by units of n bits comprises residue table storage means which stores a residue table containing 2.sup.n pieces of residue data having n/2 bits obtained by division of 2.sup.n pieces of data having n bits by the generator polynomial, data read means which reads out the data having predetermined number of bits for cyclic redundancy check by units of n bits, address generation means which generates addresses of the residue data to be read out from the residue table storage means, residue data read means which reads the residue data out of the residue table storage means according to the generated address, data save means which saves the data subjected to the cyclic redundancy check, data shift means which shifts the data subjected to the cyclic redundancy check, data synthesis means to synthesize the shifted and saved data and the residue data read out and check means for parity check of the data subjected to the cyclic redundancy checking using the residue data finally read out.

Brief Summary Text (32):
According to a further preferred embodiment of a cyclic redundancy check processor, the data save means generates save data by saving the lower n/2 bits of the data subjected to cyclic redundancy check, the data shift means generates the shift data by shifting the higher n/2 bits of the data subjected to the cyclic redundancy check toward the lower n/2 bits, and the data synthesis means generates synthetic data by synthesizing the residue data and the shift data or the residue data and the save data.

Brief Summary Text (33):
According to another preferred embodiment of a cyclic redundancy check processor, the address generation means uses, for the first address generation, n bit data subjected to cyclic redundancy check read out by the data read means in order to generate the address for the residue table, and for the following address generations, the data obtained by synthesis of the residue data having n/2 bits previously read out and n/2 bits among the n bits subjected to the cyclic redundancy check in order to generate the address for the residue table.

Brief Summary Text (34):
According to still another preferred embodiment of a cyclic redundancy check processor, the address generation means uses, for the first and following address generation processes, the data obtained by synthesis with treating the residue data of n/2 bits previously read out as the higher bits and n/2 bits of the n bits subjected to the cyclic redundancy check as the lower bits in order to generate the address for the residue table.

Brief Summary Text (35):
According to a further preferred embodiment of a cyclic redundancy check processor, the residue table has 2.sup.n pieces of residue data each having a length of n/2 bits obtained by modulo 2 divisions with the generator polynomial G (X) of the data polynomials ##EQU3## for 2.sup.n pieces of data with groups of n bits {Dk} (Dk= (d.sub.0, . . . ,d.sub.i, . . . , d.sub.n-1) where di is 0 or 1, i is an integer satisfying 0.ltoreq.i.ltoreq.n-1 and k is an integer satisfying 0.ltoreq.k.ltoreq.2.sup.n.

Brief Summary Text (36):
According to still another preferred embodiment of a cyclic redundancy check processor, the residue table has, if n/2-m=0, 2.sup.n pieces of residue data {r.sub.0, . . . , r.sub.m-1 }, each having a length of n/2 bits, obtained by modulo 2 divisions with the generator polynomial G(X) of the data polynomials ##EQU4## for the data {Dk} and, if n/2-m<0, n/2 bit data comprising 2.sup.n pieces of m bit

residue data {r.sub.0, . . . , r.sub.m-1 } obtained by modulo 2 divisions with the generator polynomial G(X) of the data polynomials and the remaining n/2-m bit data of the data {Dk}, which are {d.sub.0, . . . , d.sub.n/2-m+1 }.

Detailed Description Text (4):
FIG. 1 is a block diagram to illustrate the function of the CPU 100. In the figure, the CPU 100 comprises a data reader 101 to read out the data to be processed from the data memory 300, a reference address generator 102 to generate addresses for the residue data to be read out of the residue table TR in the table ROM 200, a residue table reader 103 to read out the residue data from the residue table TR in the table ROM 200, a save data generator 104 to save the data read out in the cyclic coding and CRC processing, a shift data generator 105 to shift the data read out in the cyclic coding and CRC processing, a synthetic data generator 106 for data synthesis in the cyclic coding and CRC processing, a repetition controller 107 to control repetition of the cyclic coding and CRC processing and registers 110 to be used for data operation in the cyclic coding and CRC processing.

Detailed Description Text (9):
Though the computer explained above executes operation, data processing and control by units of 8 bits, the description above can be applied to computers for n-bit processing in general. Suppose a generator polynomial G(X) of degree m (m is an integer satisfying m .ltoreq.n/2) is used in cyclic coding and cyclic redundancy code check for predetermined number of data, or in other words, data bits in the data memory 300 are read out by units of n bits for the above processing. In this case, the residue table can be defined as follows.

CLAIMS:

6. A cyclic redundancy check processor for a generator polynomial G(X) of degree m (m is an integer satisfying m.ltoreq.n/2) using a computer for data processing by units of n bits comprising:

residue table storage means for storing a residue table containing 2.sup.n pieces of residue data having n/2 bits obtained by division of 2.sup.n pieces of data having n bits by said generator polynomial;

data read means for reading out said data having n bits for cyclic redundancy check;

address generation means for generating addresses of said residue data to be read out from said residue table storage means;

residue data read means for reading said residue data out of said residue table storage means according to the generated addresses,

data save means for saving the data for said cyclic redundancy check, said data save means generating save data by saving the lower n/2 bits of the data for said cyclic redundancy check;

data shift means which shifts the data for said cyclic redundancy check, said data shift means generating the shift data by shifting the higher n/2 bits of the data for said cyclic redundancy check toward the lower n/2 bits;

data synthesis means for synthesizing said shifted and saved data and said residue data read out, said data synthesis means generating synthetic data by synthesizing said reside data with one of said shift data and said save data; and

check means for parity check of the data for said cyclic redundancy checking using said residue data finally read out.

7. A cyclic redundancy check processor as set forth in claim 6 wherein said address generation means uses, for a first address generation, n bit data for said cyclic redundancy check read out by said data read means in order to generate said first address for said residue table, and for each of the following address generations, the data obtained by synthesis of the residue data having n/2 bits previously read out and n/2 bits of the n bit data for said cyclic redundancy check in order to generate said each of the following address for said residue table.

9. A cyclic redundancy check processor as set forth in claim 6 wherein said residue table has 2.sup.n pieces of residue data each having a length of n/2 bits obtained by modulo 2 divisions with said generator polynomial G (X) of the data polynomials ##EQU9## for 2.sup.n pieces of data with groups of n bits {Dk} (Dk=(d.sub.0, . . . , d.sub.i, . . . , d.sub.n-1) where di is 0 or 1, i is an integer satisfying 0.ltoreq.i.ltoreq.n-1 and k is an integer satisfying 0.ltoreq.k.ltoreq.2.sup.n.

10. A cyclic redundancy check processor as set forth in claim 9 wherein said residue table has, if n/2-m=0, 2.sup.n pieces of residue data {r.sub.0, . . . , r.sub.m-1 }, each having a length of n/2 bits, obtained by modulo 2 divisions with said generator polynomial G(X) of the data polynomials ##EQU10## for said data {Dk} and, if n/2-m>0, n/2 bit data comprising 2.sup.n pieces of m bit residue data {r.sub.0, . . . , r.sub.m-1 } obtained by modulo 2 divisions with said generator polynomial G(X) of said data polynomials and the remaining n/2-m bit data of said data {Dk}, which are {d.sub.0, . . . , d.sub.n/2-m+1 }.

Previous Doc Next Doc Go to Doc#

L7: Entry 1 of 7 File: USPT Dec 28, 2004

DOCUMENT-IDENTIFIER: US 6836869 B1
TITLE: Combined cyclic redundancy check (CRC) and Reed-Solomon (RS) error checking unit

Abstract Text (1):
An error checking circuit that performs RS encoding and decoding operations and also generates CRC codes includes a configurable two-stage combinatorial circuit that carries out selected finite-field arithmetic operations associated with RS and CRC coding. Input registers store the generator polynomial and operand coefficients associated with the data blocks or packets being encoded or decoded, and an output register holds the intermediate working result and at the end the final result of the finite-field arithmetic operation. Each stage of the combinatorial circuit includes sets of AND and XOR gates performing bitwise finite-field multiply and add on the operand bits, and the connections between registers and gates and between gates in the two stages are configured by multiplexer units responsive to RS and CRC instructions. The two-stage combinatorial block can be replicated into a 4-stage or 8-stage arithmetic circuit for CRC mode.

Brief Summary Text (2):
The present invention relates to error detection and correction circuitry for reliable digital data storage or transmission. In particular, the invention relates to circuitry that employs finite (Galois) field polynomial arithmetic for generating error checking codes, such as Reed-Solomon (RS) codes and cyclic redundancy check (CRC) codes, or for decoding data containing such codes.

Brief Summary Text (4):
Error checking codes have been developed to assure digital data integrity in situations such as transmission of data over a communications channel or data storage onto and retrieval from a storage medium. The codes allow the existence of errors in the data stream to be identified and in most cases corrected. Two widely used error correction codes are Reed-Solomon (RS) and cyclic redundancy check (CRC) codes. They are often used together, with RS codes being used for every block of 255 bytes of an encoded word, and CRC codes being used cumulatively for every packet of 10 or 20 RS encoded blocks. Typically, distinct circuitry is used to generate the respective RS and CRC codes, although the respective circuitry can be largely operated in parallel.

Brief Summary Text (5):
In U.S. Pat. No. 5,383,204, Gibbs et al. describe an error correction system having distinct RS encoding and CRC encoding hardware units sharing the same codeword memory and processing the data largely in parallel. The CRC check bytes are calculated on a data packet, while the RS processing is done for each data block, or in the case of the final four blocks of a packet, for the data blocks plus CRC check bytes. Since the data that the two units encode is the same, only the RS encoding of the CRC check bytes appended to the final four blocks is not parallizable. Once the CRC check bytes have been calculated, the RS bytes can be calculated for those final blocks. The patent deals with how the codeword memory accesses are sequenced under the operation of the two encoding units.

Brief Summary Text (6):

In U.S. Pat. No. 5,671,237, Zook describes a two-stage CRC encoding/decoding system. The first stage takes the input data, encodes it, and puts both the input data and K.sub.1 CRC bytes in memory. Then the second stage (with K.sub.2 >K.sub.1 streams) takes the input data plus K1 CRC bytes from the memory, verifies that no errors were introduced by the memory, encodes it, and passes the data plus K.sub.2 CRC bytes as output from the unit. Each CRC stage (a generator/checker) uses programmable 8-bit generator polynomials. Each stage has a segmenter, which splits a data input stream into a programmable number of substreams, and has a programmable variable number (m) of parallel CRC units, each operating on a separate data substream. The first byte goes to a first CRC unit, the second byte to a second CRC unit, the m.sup.th byte to an m.sup.th CRC unit, the (m+1)-st byte to the first CRC unit, etc., until the end of the input data. The error checking capability of the CRC system, usually limited to detecting errors lying entirely in one 8-bit block, is enhanced by multiple staging and by interleaving.

Drawing Description Text (2):
FIG. 1 is a schematic block diagram of a combined RS/CRC byte-processing unit in accord with the present invention for both RS encoding and decoding and CRC check code generation.

Detailed Description Text (3):
With reference to FIG. 1, in order for the combined Reed-Solomon (RS)--Cyclic Redundancy Check (CRC) unit to use the same processing circuitry for RS encode and decode and also for CRC generation, the unit has two distinct modes of operation, an RS mode and a CRC mode, configured by multiplexers 33 in response to received instructions. The unit operates in its RS mode when executing RSEAC (RS encode) and RSDAC (RS decode) instructions. The unit operates in its CRC mode when executing any of several CRC instructions (CRC32, CRC16, CRC8, CRX32, CRX16, CRX8), which also specify 8-bit, 16-bit and 32-bit operation.

Detailed Description Text (4):
The unit has five 32-bit registers 11, 13, 15, 17, and 19: four input registers (A, B, C and P) and one output register (RSD). Registers A and C are shift registers controlled by a shift clock generated by a 4-bit clock counter 23 responsive to a start signal when beginning an RS or CRC instruction. In the embodiment of FIG. 1, register A shifts by two bits per clock in CRC mode. (In alternative embodiments, detailed below with reference to FIGS. 6 to 9, the register A shifts by four or eight bits per clock in CRC mode.) The register RSD is a result register that has data latched into it on a clock edge. Register P is loaded with a generator polynomial for the particular RS or CRC operation. Register B is loaded with a coefficient in RS mode or with a copy of the generator polynomial in CRC mode. The generator polynomial loaded into register P (and register B in CRC mode) may be 8-bits, 16-bits or 32-bits, with a (most significant) 9.sup.th, 17.sup.th or 33.sup.rd bit of the generator not stored, but simply implied, as it is always one. 8-bit or 16-bit generators are packed into the most significant 8 or 16 bit locations in the register, in which case the lower 24 or 16 bits of the register are unused.

Detailed Description Text (14):
The A input buffer 10 loads 4 bytes (32 bits) of an input data stream into the A shift register 11. The A register 11 shifts two bits per clock in the CRC mode, transferring bits A31 and A30 out of the register to respective first and second stages of the combinatorial circuitry 25 in response to each clock pulse supplied by the 4-bit clock counter 23. The unit's C register 13 is not used in CRC mode. The B and P registers 15 and 17 are loaded with identical copies of the generator polynomial in response to a RSPB instruction. The P register contents P31 to P0 are used by the first stage of combinatorial circuitry 25, while the B register contents B31 to B0 are used by the second stage. The working contents D31 to D0 in the result register RSD are also used in feedback via a bus 31 to the first stage XOR gates 29-1.

Detailed Description Text (15):
Because the CRC generation can use 32-bit, 16-bit or 8-bit generator polynomials (packed into the most significant end of the B and P registers if less than 32-bits), the circuitry 25 includes configuration multiplexers 39-42 for setting up operation with the smaller generators. In particular, if an 8-bit generator is used, then multiplexers 41 and 42 select the two transferred bits A31 and A30 from the A register for supply to the respective first and second stage XOR gates 29-1 and 29-2 corresponding to the bit 24 location. If a 16-bit generator is used, then multiplexers 39 and 40 select those transferred bits A31 and A30 for supply to the XOR gates corresponding to the bit 16 location. If a 32-bit generator is used, the bits A31 and A30 are supplied to the XOR gates corresponding to the bit 0 location. The other first stage XOR gates 29-1 receive feedback bits D31 to D25, D31 to D17 or D31 to D1, depending on the generator size. The other second stage XOR gates 29-2 receive first stage XOR gate outputs corresponding to bits 30 to 24, 30 to 16 or 30 to 0, again depending on the generator size. (Actually, all gates have inputs regardless of generator size, but since the generator values in the unused bits of registers P and B for small order generators are zero, the corresponding register RSD values for the unused bits always remain zero.)

Detailed Description Text (29):
The finite field operation p(x)=i(x).multidot.x.sup.n-k mod g(x) can be performed as a series of multiply and accumulate steps, as described below. When configured in RS mode, the RS/CRC unit performs a Galois-Field multiply-accumulate operation (GFMAC) in the field GF(256). Referring to the operands by the registers in which they are stored, the particular GFMAC operation is RSD=((A*B)+C) mod P, where each of the operands A, B and C is a byte regarded as a polynomial of order 7 or less over GF(2), and P is an irreducible generator polynomial of order 8 over GF(2). This is an 8-bit operation. Partial results are saved in the register RSD after each step until the RS operation is completed.

Detailed Description Text (30):
The unit has five 32-bit registers, each holding four bytes at any one time. These are the same registers that are used for CRC code generation when the unit is in the CRC mode. Each of the four bytes in a register operates independently and in parallel with the other three bytes in that register. In the RS mode, the unit can process four sets of bytes in parallel, performing 4 independent multiply-accumulate operations simultaneously. It takes 8 clock cycles to complete the 4 parallel multiply-accumulate operations. The registers are used as follows in RS mode:

Detailed Description Text (33):
Both RSEAC and RSDAC perform the GFMAC operation RSD=((A*B)+C) mod P on the 4 sets of 1-byte operands. The P register is loaded with the generator polynomial by a RSPB instruction. (The B register is loaded also, but is subsequently overwritten with operand data. Unlike the CRC mode described below, the B register has a different function in RS mode.) For both RSEAC and RSDAC instructions the polynomial P is loaded into the least significant byte of the P register, and that byte is broadcast to all 4 GFMAC units; the other 3 bytes in the P register are ignored.

Detailed Description Text (55):
The RS/CRC unit is capable of fast calculation of CRCs on both incoming and outgoing data streams. The unit supports standards, which specify 8-bit, 16-bit or 32-bit generator polynomials that will respectively produce an 8-bit, 16-bit or 32-bit CRC. Arbitrary packet sizes are supported. The initial value of the CRC before calculation can be set to 0 (the normal case), all 1's (which is required in many standards) or any other value specified by the particular CRC standard.

Detailed Description Text (56):

When configured in CRC mode, the unit behaves as a polynomial long division unit, calculating a CRC by dividing a data packet, treated as a polynomial, by a generator polynomial. A data packet X of length n bytes is treated as a polynomial of order 8*n-1 over GF(2). The generator polynomial P is a selected monic polynomial of order 8, 16 or 32 over GF(2) with the most significant bit being implied. The CRC unit calculates the remainder after division of X by P. The remainder after division is the CRC code for the data packet. The unit has A, P, B and RSD registers, for input data stream, generator polynomial, generator polynomial and CRC result respectively. These registers are the same as those used by the unit in Reed-Solomon (RS) mode.

Detailed Description Text (59):
In order to use the RS/CRC unit in CRC mode, you must first set up the P and B registers with the CRC generator polynomial using the RSPB instruction. RSPB, which is executed once at the beginning of the calculation, loads 2 identical copies of the generator polynomial which have been previously set up in ra and rb. It loads one copy into the P register and one copy into the B register of the CRC unit. The P and B registers are 32-bit registers. The generator polynomial has order 8, 16 or 32 which means it is 9, 17 or 33 bits long, but since the most significant bit is always 1, we need only load the least significant 8, 16 or 32 bits of the polynomial into the P and B registers. The polynomial is packed into the most significant bit end of the P and B registers. In the case of 16 and 8 bit polynomials, the lower 16 or 24 bits of the P and B registers are unused.

Detailed Description Text (60):
CRC generator polynomials are primitive over GF(2). A common 32-bit generator CRC-32 is: x.sup.32 +x.sup.26 +x.sup.23 +x.sup.22 +x.sup.16 +x.sup.12 +x.sup.11 +x.sup.10 +x.sup.8 +x.sup.7 +x.sup.5 +x.sup.4 +x.sup.2 +x+1, represented in the registers P and B by the (truncated) hexadecimal value 04C11DB7. There are several common 16-bit generators, including x.sup.16 +x.sup.12 +x.sup.5 +1 (CRC-CCITT), represented in the registers by the hexadecimal value 1021, and x.sup.16 +x.sup.15 +x.sup.2 +1 (CRC-16), represented by the hexadecimal value 8005. A common 8-bit generator is x.sup.8 +x.sup.4 +x.sup.3 +x.sup.2 +1, represented by the hexadecimal value 1D. Other primitive polynomials of order 8, 16, or 32 in the field GF(2) could be used. (E.g, there are a total of 16 primitive polynomials of order 8, and 2048 of order 16.)

Detailed Description Text (61):
Although the present preferred embodiment employs and common standards specify generators of order 8, 16, and 32, the generators need not have an order of 2.sup.n. For example, a generator of order 12 or 20 is possible. At some future point in time, even larger order generators, e.g. of order 40, 48, or 64, may be specified in a CRC standard.

Detailed Description Text (64):
You use a CRC32/16/8 instruction depending on whether the generator polynomial is 32/16/8 bits long. CRC32/16/8 is executed n/4 times, where n is the packet size in bytes. CRC32/16/8 loads the next 4 bytes of the input data stream from ra into the CRC unit's A register. Only ra is used. As the data is transferred from ra to the A register, it passes through the input bit-endian and byte-endian muxes. (Some CRC standards specify that input data bytes be reflected or bits be inverted prior to processing, while other standards process the packet bytes directly.) Once in the A register, data is processed with bit 31 as the most significant bit (msb) and bit 0 as the least significant bit (lsb). rb is altered as a side effect. Following the load, the CRC instruction processes the 32-bit word one byte at a time at a rate of 2 bits per clock cycle, so that processing each input word takes 16 cycles.

Detailed Description Text (65):
The A register is preferably double-buffered so that the CRC unit can be utilized every cycle. Since CRC operations take many clock cycles (16), you can issue a

second CRC instruction immediately after issuing a first one, and both calculations will proceed to completion. The unit buffers the A input value for the second instruction while the first instruction is running. When the first instruction completes, the second one begins immediately. However, if a third CRC32/16/8 instruction is attempted before the first instruction is completed and when both buffers are full, the third instruction will stall until the first instruction is complete and the unit can start another.

Detailed Description Text (67):
All of these instructions process one byte (8 bits) of data at a time. They are used for processing the last 1-3 bytes of a packet whose size is not an exact multiple of 4. You use a CRX32/16/8 instruction depending on whether the generator polynomial is 32/16/8 bits long. CRX32/16/8 is executed n mod 4 times, where n is the packet size in bytes. CRX32/16/8 loads the next byte of the input data stream from ra into the CRC unit's A register. Only ra is used. As the data is transferred from ra to the A register, it passes through the input bit-endian and byte-endian muxes. Once in the A register, one byte of data is processed with bit 31 as the most significant bit and bit 24 as the least significant bit. rb is altered as a side effect. Following the load, the CRX instruction processes 1 byte in 1 cycle.

Detailed Description Text (69):
MOV reads the CRC result from the D register. The CRC result is in bits 31:0, 31:16, or 31:24 of the D register depending on whether the generator polynomial size is 32, 16 or 8. As the data is transferred from D to the ra register, it passes through the output bit-endian and byte-endian muxes. (Some CRC standards specify that the CRC code bytes be reflected or their bits be inverted, while others use the value taken directly from the result register.)

Detailed Description Text (72):
In FIG. 6, the A input buffer 60 loads 4 bytes (32 bits) of an input data stream into the A shift register 61. The A register 61 in the CRC mode shifts as many individual bits per clock as there are stages (e.g, bits A31 to A28 for a 4-stage circuit, or bits A31 to A24 for an 8-stage circuit). That is, as seen in FIGS. 6 and 7, the first stage 65-1 receives bit A31, the second stage 65-2 receives bit A30, the third and fourth stages in pair 67-B receive respective bits A29 and A28, and the fifth through eighth stages in pairs 67-C and 67-D, if present, receive respective bits A27 through A24. The A register 61 continues to shift only one bit per clock cycle in the RS mode. The unit's C register is not used in CRC mode. Per the previous embodiment in FIGS. 1 and 4, the B and P registers are loaded with identical copies of the generator polynomial in response to a RSPB instruction. The P register contents, P31 to P0, are used by the odd-numbered stages (the first stage in each successive pair of stages), while the B register contents, B31 to B0, are used by the even-numbered stages (the second stage in each successive pair of stages). Alternatively, the P register could be used by all stages, and the B register left unused in CRC mode. The working contents D31 to D0 in the RSD result register 62 are also used in feedback via a bus 63 leading back to the first combinatorial stage of the arithmetic circuit.

Detailed Description Text (73):
As before, because the CRC generation can use 32-bit, 16-bit or 8-bit generator polynomials (packed into the most significant end of the P and B registers if less than 32-bits), the combinatorial stages 65-1, 65-2, etc. include configuration multiplexers 69 and 70 for setting up operation with smaller generators, as seen in FIGS. 6 and 8. If an 8-bit generator is used, then multiplexers 70 supply the transferred bits from the A shift register 61 (respective bits A31 to A24, depending on the particular stage) to the XOR gates at the bit 24 location of each stage. If a 16-bit generator is used, then multiplexers 69 supply the transferred bits from the A register to the XOR gates at the bit 16 location of each stage. However, if a 32-bit generator is used, then the transferred bits from the A shift register 61 are supplied to the XOR gates corresponding to the bit 0 location. All

other XOR gates receive previous stage XOR gate outputs, or feedback bits from result register RSD for the first stage XOR gates, of corresponding bit locations shifted by one, as seen in FIGS. 6 and 8. That is, the inter-stage connection for the XOR gates is shifted by one, as seen in FIGS. 6 and 8, so that XOR stage outputs T0 to T30 (and feedback bits D0 to D30) that correspond to bit locations 0 to 30 are supplied to the next stage (or first stage) XOR gates at respective bit locations 1 to 31. (Note: Since the result register contents are initially all zero, and any unused generator polynomial bits P0 to P15 and B0 to B15 for 16-bit generators, or P0 to P23 and B0 to B23 for 8-bit generators, are also all zero, the lower significance bit locations in those cases will remain zero throughout the operation.)

CLAIMS:

4. The circuit of claim 1 wherein at least the input register P stores a selected generator polynomial associated with said RS and CRC codes.

5. The circuit of claim 1 wherein input register B stores a multiply coefficient in RS mode and a copy of a selected generator polynomial in CRC mode.

10. The circuit of claim 1 wherein said arithmetic circuit has multiplexer means responsive to CRC instructions indicative of generator polynomial size for supplying A register shift bits to combinatorial elements within the arithmetic circuit that are associated with a bit location associated with the indicated generator size.

Previous Doc Next Doc Go to Doc#